UNIVERSITI PUTRA MALAYSIA

MIDDLEWARE COMPONENT USING ENTERPRISE JAVA BEAN (EJB)

KAVITHA A/P KANNAN

FSKTM 2003 17

MIDDLEWARE COMPONENT USING ENTERPRISE JAVA BEAN (EJB)

KAVITHA A/P KANNAN

MASTER OF SCIENCE UNIVERSITI PUTRA MALAYSIA
MAY 2003

MIDDLEWARE COMPONENT USING ENTERPRISE JAVA BEAN (EJB)

By
KAVITHA A/P KANNAN

Thesis Submitted to the School of Graduate Studies, Universiti Putra Malaysia, in Fulfillment of the Requirements for the Degree of Master of Science in the Faculty of Computer Science and Information Technology

May 2003

TABLE OF CONTENTS

DEDICATION

To my loving late mother Adlechumit Gopal ,

beloved sister Kalavathy Kannan ,

and dearest husband Thangga Thurai Ramiah

ABSTRAK

Abstrak disertasi yang diserahkan kepada Senat Universiti Putra Malaysia bagi memenuhi keperluan untuk ijazah Master Sains

KOMPONEN MIDDLEWARE DENGAN MENGGUNAKAN ENTERPRISE JAVA BEAN (EJB)

Oleh

KAVITHA A/P KANNAN

OKTOBER 2003

Pengerusi: Puan Sazlinah bte Hasan

Fakulti: Sains Komputer and Technology Maklumat

Pembangunan perisian berasaskan komponen menjanjikan pengurangan pembangunan perisian dan kos penyelengaraan. Ini juga dapat membantu meningkatkan kemudahan pengunaan semula perisian. Komponen merupakan sekumpulan perisian yang menyediakan kemudahan seperti pengesahan *login*, transaksi pengguna dan formula pengiraan [3]. Ia adalah tersembunyi atau transparen, di mana hanya boleh akses melalui *interface* apabila digabungkan dengan sistem lain [3]. Kebanyakan kriteria ini diwarisi dari pembangunan berasaskan objek(OO).

Oleh yang demikian, laporan ini menyentuh *overview* dan pembangunan perisian berasaskan komponen menggunakan ***J2EE framework*** iaitu *Enterprise Java Beans (EJB)*. Objektif projek ini adalah untuk menampilkan penggunaan konsep

middleware di mana struktur atau reka bentuk ini membolehkan komunikasi antara lapisan antramuka dan *bussines logic* adalah tidak saling bersandaran.

Di samping itu, ia juga dapat membantu penggunaan semula komponen *enterprise bean* dan membantu menentukan piawai reka bentuk dalam pambangunan perisian dapat di capai [10]. Projek ini menggunakan dua pelanggan dihubungkan dengan satu komponen atau lapisan middleware, yang berada di dalam kontena *J2EE*. Projek ini menekankan reka bentuk dan pembangunan komponen *middleware* ini berserta dengan pengujian komponen menggunakan perisian kedai buku. Berserta ini, projek ini juga melampirkan kelebihan menggunakan *EJB* berbanding komponen model teragih dan keberkesanannya untuk digunakan atas *platform* dan bahasa pengaturcaraan yang berlainan.

ABSTRACT

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in Fulfilment of the requirement for the degree of Master of Science

MIDDLEWARE COMPONENT USING ENTERPRISE JAVA BEAN (EJB)

By

KAVITHA D/O KANNAN

OCTOBER 2003

Chairman: Puan Sazlinah bte Hasan

Faculty: Computer Science and Information Technology

Component-based software development promises to decrease software development and maintenance costs and also providing sophisticated facilities for software reuse. A component is a chunk of software that provides functionality such as login validation, customer transaction and calculation [3]. It is encapsulated which means, that it can only be accessed via its interface and can be combined with other systems [3]. Most of these characteristics are derived from object oriented development.

Therefore this paper presents an overview and development of component based software using J2EE framework called Enterprise Java Bean (EJB). The aim of this project is at capturing the middleware concept using EJB where this architecture provides loose coupling of presentation layer and business logic. Moreover, leads Having two different clients connecting to a single middleware component which

resides in the container of J2EE Server. The details about design and development of this middleware component and testing this component on to a test bed called bookstore application is the heart of this project. This paper will also introduce about the advantage of EJB against distributed component model and the interoperability of this middleware component across different platform and programming languages.

ACKNOWLEDGEMENT

I would like to take this opportunity to express my sincere gratitude to my Project Supervisor, Pn.Sazlinah Bte Hasan. It was an honor to have such a dedicated person guide and supervise me through this entire project.

My sincere appreciation must also be expressed to Dr. Mohamad Othman and Dr. Shamala for their worthy advice.

This project would not have been a success if not for the support and encouragement given by my family and friends.

Finally, I am also grateful to all those who have directly or indirectly made this thesis a success factor. I do appreciate the time and energy spent by all in helping me out throughout this project.

KAVITHA KANNAN

OCTOBER 2003

APPROVAL SHEET

This thesis was submitted to the faculty of Computer Science and Information Technology of Universiti Putra Malaysia has been accepted as fulfillment of the requirements for the degree of **Master of Science.**

Puan Sazlinah bte Hasan

Lecturer

SAZLINAH BT. HASAN
Pensyarah
Jabatan Teknologi Komunikasi dan Rangkaian
Fakulti Sains Komputer dan Teknologi Maklumat
Universiti Putra Malaysia
43400 UPM Serdang, Selangor

Faculty of Computer Science and Information Technology

Universiti Putra Malaysia

Date:

DECLARATION

I hereby declare that the project is based on my original work for quotations and citations, which have been duly, acknowledge. I also declare that it has not been previously or concurrently submitted for any other degree at UPM or other institutions.

KAVITHA D/O KANNAN

Date: 14/11/03

LIST OF FIGURES

LIST GLOSSARY OF TERMS

Application server

A server program that allows the installation of application specific software components, in a manner so that they can be remotely invoked, usually by some form of remote object method call.

Bean-managed persistence

When an Enterprise JavaBean performs its own long-term state management.

Bytecode

In the context of Java, bytecode is the platform-independent executable program code.

Component standard

A definition of how software components cooperate, and in particular the roles and interfaces of each. In the context of Java middleware, component standards usually include specifications of the middleware interfaces exposed to the components, and the component interfaces required by the middleware.

Container managed persistence

When an Enterprise JavaBean server manages a bean's long-term state.

CORBA

Standard maintained by the Object Management Group (OMG), called the Common Object Request Broker Architecture.

COS Naming

CORBA standard for object directories.

Data source

This is the term used by the JTA and JDBC specifications to refer to persistent repository of data. It usually represents a database. It also may refer to an object that makes database connections available (i.e. a driver).

DCOM

Microsoft's Distributed Component Object Model.

Enterprise JavaBeans (EJB)

A server component standard developed by Sun Microsystems.

Entity bean

An Enterprise JavaBean that maintains state across sessions, and may be looked up in an object directory by its key value.

IDL

interface description language, CORBA's syntax for defining object remote interfaces.

IIOP

Internet Inter-ORB Protocol, CORBA's wire protocol for transmitting remote object method invocations.

Java Naming and Directory Interface (JNDI)

The Java standard API for accessing directory services, such as LDAP, COS Naming, and others.

JVM

Java virtual machine.

Middleware

Software that runs on a server, and acts as either an application processing gateway or a routing bridge between remote clients and data sources or other servers, or any combination of these.

OMG

Object Management Group, an organization that defines and promotes object oriented programming standards.

ORB

object request broker, the primary message routing component in a CORBA product.

Persistence

Maintaining state over a long time, especially across sessions.

RMI

Remote Method Invocation, the Java standard technology for building distributed objects whose methods can be invoked remotely across a network.

RMI over IIOP

Using the CORBA IIOP wire protocol from an RMI API.

Servlet

An application extension to a Java Web server.

Session bean

An Enterprise JavaBean that does not maintain its state from one session to the next. Appears to the client as if the bean was created just for that client.

Skeleton

A server-side software component that serves to relay remote calls from a client to the methods of a servant running in a server. Usually a skeleton is automatically generated by a special compiler.

Stub

A client-side software component that serves to forward remote calls to a remote server, and receive the subsequent responses. Usually automatically generated by a special compiler.

Three-tier

An architecture in which a remote client accesses remote data sources via an intervening server.

CHAPTER 1

INTRODUCTION

To support rapid software development, applications are currently constructed from reusable components. Using this approach, the architecture of an application can be considered as a collection of interconnected components, usually in a distributed environment. The Sun Microsystem's Enterprise Java Beans (EJB) version 2.0 is one of the currently used, component based platforms for development of distributed object-oriented applications.

Today there many sufficient software companies that are trying to compete in developing middleware component. The design and development of this component is not as easy as it projects. Information sharing with high security and authentication need to be imposed on this component in order to avoid miss use of information, which are confidential to certain organization. This can be seen in authentication process where user proves his or her identity to a system by giving the user login and password to access the system. While another process is called authorization where the J2EE server grants or denies permission to access a resource such as the enterprise bean components which is highly confidential to the organization [1]. Other then that authorized application must be able to access this information across network (either local or remote) and must be accurate at any point of time.

Reuse of component will lead to sharing of information pertaining to a business. Any software written must be able to run on network where the functionality component is distributed in a remote machine. Platform independency is on the

leading issue in Java based component development; they support multiple inheritances and are platform independent, as long as a Java Virtual Machine (VM) exists. This increases the cohesion within the components and decreases coupling between them.

1.1 Middleware Component

The motivation for middleware component models partly originates from deficiencies present in available Object Oriented middleware technologies. Designs and implementations of applications using middleware invariably have a large focus on middleware, which can cause a distraction from the main business problem at hand. Moreover experience has shown that integrating existing middleware technologies is cumbersome and often a source of additional complexity [7].

Distributed component technologies combine the characteristics of components with the functionality of middleware systems to provide inter-process communication between components [7]. That is to say components that can communicate across machine boundaries. EJB components model is a server-side component model used for developing distributed business objects also addressed as middleware component. These are used on the middle tier application servers that manage the components at runtime and make them available to remote clients. The EJB components or middleware operate by providing components with a container in which they can execute. Each component provides an interface used for life-cycle operations such as creation, migration and destruction, as well as the remote interface it supports. Containers themselves run on application servers,